Aprende LOGO con AMSTRAD

Ficheros en castellano



ra-ma

APRENDE LOGO CON AMSTRAD

FICHEROS EN CASTELLANO

ra-ma editorial
CARRETERA DE CANILLAS, 144 - 28043 MADRID
TELEFS (91) 200 97 46/47

Centro Educativo de Informática.
Autor: F. Javier Alvarez.
Coordinación: Saturnino Hernández.

Editado por RA-MA
Carretera de Canillas, 144
28043 MADRID
España.

Imprime: Signo Impresores, S. A.
Albasanz, 27. 28037 Madrid.
Depósito Legal: M. 11186 - 1986
I. S. B. N.: 84 - 86381 - 16 - 8

Composición: A. Andeyro.

"La Geometría de la Tortuga es una ayuda para aprender otras cosas porque estimula el uso consciente y deliberado de estrategias de resolución de problemas y estrategias matemáticas".

Seymour Papert

INSTRUCCIONES PARA CARGAR EL PROGRAMA LOGO

1. Introducir el disco de LOGO, y teclear:

   ¦CPM [RETURN]

2. Cuando aparezca un signo de interrogación en pantalla, teclear:

   LOAD "CAST [RETURN]

INDICE

INTRODUCCION

El prodigioso avance de la microinformática en los últimos años ha sido la causa de que el ordenador haya penetrado en múltiples actividades de la vida humana y entre ellas, y de modo especial, en la didáctica.

Las posibilidades que ofrecen los ordenadores actuales en cálculo, gráficos, colores y sonidos permiten simular un mundo que se rige por las mismas leyes que la Naturaleza y donde se puede sustituir la observación y el conocimiento empírico allá donde no es factible que éste llegue.

Entre los innumerables lenguajes de programación existentes, cada uno con una orientación específica, LOGO destaca por su carácter pedagógico. Su creador, Seymour Papert, procedente del campo de la psicología genética, desarrolló en el Instituto Tecnológico de Massachusetts (MIT), al frente de un equipo de colaboradores expertos en investigación didáctica, un lenguaje que ha supuesto una auténtica revolución en las aplicaciones del ordenador a la enseñanza (CAI).

Papert aplicó las ideas sobre aprendizaje cognitivo del epistemólogo Piaget basadas en que el mejor conocimiento es el obtenido de una forma natural, sin imposiciones, tal como se aprende el idioma materno o se adquiere el concepto de espacio. De esta manera, Papert diseña un lenguaje donde el niño, una vez aprendidas las reglas fundamentales del lenguaje, es introducido en un entorno ambiental, en un microcosmos en palabras del autor, donde irá adquiriendo de una forma afectiva múltiples conceptos abstractos que han dejado de serlo para convertirse en ideas concretas. Cuando un niño, en el entorno LOGO, construye un triángulo ha descubierto por sí mismo que ha tenido que sumar 180 grados para lograr cerrar con tres líneas una superficie.

Se ha eliminado, en este momento, el aprendizaje disociado en la enseñanza tradicional de dos conceptos relacionados:

a) qué es un triángulo
b) suma de los lados de un triángulo

En el ambiente LOGO los dos conceptos se funden en un único conocimiento que los engloba.

El papel que desempeña el profesor en un ambiente LOGO es el de orientar y canalizar las experiencias de los niños para conseguir el ulterior aprendizaje que viene dado como conclusiones del niño a la actividad experimental que ha desarrollado.

LOGO, como lenguaje de programación, se presenta como un lenguaje sencillo, potente y perfectamente válido para adquirir conceptos de programación tales como bifurcaciones, procedimientos, listas, recursividad, etc.

Conscientes de la dificultad que puede entrañar el aprendizaje de un lenguaje con unas instrucciones en inglés para niños de temprana edad -en EEUU se han realizado experiencias positivas de aprendizaje con LOGO en niños de seis años- en este manual se explican las instrucciones del lenguaje traducidas al castellano. Con ello se ha pretendido eliminar la desnaturalización que supondría dar órdenes en un lenguaje que se desconoce y minimizar el tiempo de aprendizaje de las instrucciones básicas para que el niño esté desde un primer momento en condiciones de trabajar con el ordenador.

El autor

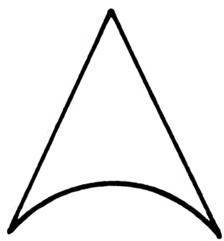
CAPITULO 1

Qué es LOGO

LOGO es un lenguaje de programación, como todo lenguaje de programación es un conjunto de órdenes y reglas sintácticas comprensibles por el ordenador.

En LOGO vamos a tener una protagonista que nos obedecerá fielmente las órdenes que le demos.

Esta protagonista es LA TORTUGA DE LOGO, rápidamente te familiarizarás con ella.



La Tortuga de LOGO

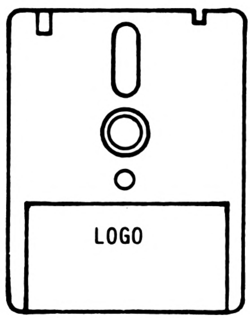
1.1. LOGO EN CASTELLANO.

Las órdenes que damos a LOGO para hacer un programa están en inglés porque el creador de este lenguaje eligió la lengua inglesa para codificar las instrucciones.

Sin embargo, para que nos sea más sencilla la utilización del LOGO, emplearemos el español que por medio de un traductor convierte las instrucciones de LOGO a inglés evitándonos tener que recordar abreviaturas inglesas.

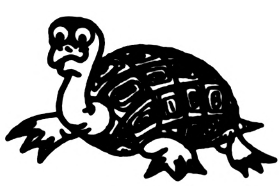
1.2. CARGA DE LOGO.

Para poder trabajar con LOGO introduciremos en el drive (unidad de disco) del ordenador, el disco de LOGO y a continuación teclearemos ¦cpm. Después de unos segundos nos saldrá en pantalla el signo "?" que es la divisa de LOGO mediante la cual se nos indica que la tortuga espera que le empecemos a dar órdenes.



1.3. CARGA DE CASTELLANO.

Si queremos dar las órdenes en castellano, tendremos que introducir el disco de castellano y escribir a continuación de la divisa load "cast y en unos instantes aparecerán órdenes traducidas al castellano con las que se puede trabajar directamente.



RESUMEN DEL CAPITULO 1

* LOGO es un lenguaje de programación.

* Utilizando las instrucciones de LOGO, podremos dirigir la tortuga para que dibuje gráficos.

* Con el traductor de castellano se pueden utilizar las instrucciones de LOGO en castellano.

* Para cargar el LOGO introduciremos el disco de LOGO en la unidad de disco y teclearemos ¦cpm

* Una vez cargado el LOGO podemos trabajar en castellano introduciendo el disco de castellano y escribiendo a continuación load "cast

ACTIVIDADES

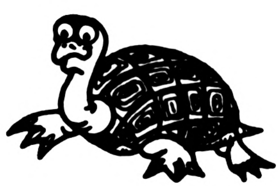
Rellena los espacios en blanco en las frases que vienen a continuación:

1. LOGO es un .... de programación.

2. Existe una .... que irá describiendo un dibujo a medida que se mueve.

3. Para cargar el LOGO introduciremos el disco de .... y teclearemos ....

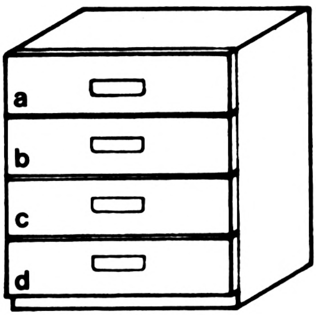
4. Para programar en castellano, después de cargar el LOGO introduciremos el disco de .... y teclearemos load "........



CAPITULO 2

Concepto de variable

Imaginemos un armario con varios cajones: A, B, C y D. Dentro de cada uno de estos cajones podemos tener objetos distintos.



En el cajón A, un libro
En el cajón B, un estuche
En el cajón C, un jersey
En el cajón D, unos zapatos

Ahora podemos cambiar lo que contiene el cajón A, sacando el libro e introduciendo en su lugar un peine. De esta manera, hemos cambiado el contenido del cajón A, pero el cajón sigue siendo el mismo. Podemos establecer un símil entre cada cajón y su contenido con su espacio en la memoria que se identifica por un nombre y donde se aloja un dato. A este espacio en memoria se le llama variable.

2.1. ASIGNACION INTERNA. INSTRUCCION "HAZ".

En LOGO el nombre de una variable va precedido de comillas (").

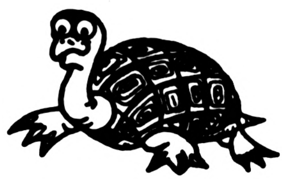
Si queremos crear una variable que tenga de nombre v y como contenido 5, escribiremos:

```
haz "v 5
```

que quiere decir: "Haz que la variable v tenga como contenido 5".

A partir de este momento, cuando hagamos referencia al contenido de la variable v escribiremos el nombre de la variable precedido de dos puntos, es decir, :v

Escribiendo ahora tras la divisa de LOGO :v aparecerá en pantalla el contenido de la variable v, es decir, 5.



EJEMPLOS:

1. Haz "número 14 ⟶ El contenido de la variable es 14.

2. Haz "nombre Javier ⟶ El contenido de la variable nombre es Javier.

3. Haz "gasto 1000 ⟶ El contenido de la variable gasto es 1000.

EJERCICIOS:

Escribe una instrucción para que una variable que se llame ciudad tenga como contenido el nombre de la ciudad en que vives.

2.2. ASIGNACION EXTERNA. INSTRUCCION "ENTRADA".

A una variable también se le puede dar un valor con la instrucción "entrada" mediante la cual introducimos un valor cuando se ejecuta un programa.

Pongamos un ejemplo:

entrada "ciudad

Con esta instrucción, LOGO esperará que le introduzcamos una ciudad determinada y el nombre de dicha ciudad será el contenido de la variable ciudad.

```
? entrada "ciudad
```

Con la instrucción entrada, LOGO espera que se le introduzca un dato.

Escribamos Madrid, por ejemplo. A continuación LOGO introduce la palabra "Madrid" como contenido de la variable ciudad y nos muestra la divisa "?".

```
? entrada "ciudad
Madrid
?
```

LOGO nos muestra la divisa "?" esperando una nueva orden.

Si queremos ver lo que contiene la variable ciudad, escribiremos :ciudad y veremos que LOGO nos contestará Madrid.

```
? :ciudad
Madrid
```

LOGO nos muestra el contenido de la variable ciudad escribiendo Madrid.

2.3. IMPRESION DE UNA VARIABLE.

El contenido de una variable puede salir reflejado en el monitor escribiendo el nombre de la variable precedido de dos puntos (:)

EJEMPLO:

```
Haz "asignatura matemáticas

:asignatura
```

Después de estas dos instrucciones el ordenador escribirá en pantalla el contenido de la variable asignatura, es decir, matemáticas.

Otro método de escribir el contenido de una variable es mediante la instrucción "escribir".

EJEMPLO:

```
Escribir :asignatura
```

El ordenador nos escribe el contenido de la variable asignatura, Madrid, de igual modo que cuando escribimos la variable precedida de dos puntos (:)

2.4. BORRADO DE UNA VARIABLE.

Para borrar una variable se utiliza la instrucción borrar.

```
? borrar :ciudad
escribir :ciudad
ciudad has no value
```

En el presente ejemplo la instrucción borrar seguida de la variable :ciudad provoca que se borre su contenido y el ordenador manda un mensaje indicando que ciudad no tiene valor.

Esto significa que ciudad no tiene valor, es decir, se ha borrado.

RESUMEN DEL CAPITULO 2

* Una variable es un espacio en la memoria que se identifica por un nombre y donde se aloja un dato.

* haz: Asigna un valor a una variable.

* entrada: Permite introducir un valor desde la consola cuando se ejecuta el programa introduciéndolo como contenido de una variable.

* borrar: Borra la variable que se especifica.

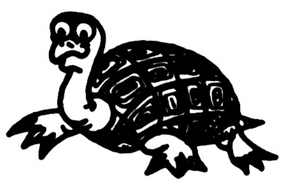
ACTIVIDADES

1. Introduce como contenido de una variable llamada flor la palabra amapola.

2. Borra la variable flor.

3. Introduce desde la consola el nombre del mes en que estamos en una variable llamada mes.

4. Introduce desde la consola el día del mes en que estamos en una variable llamada día y después escribe el contenido de esta variable.



CAPITULO 3
La tortuga de LOGO

A partir de este momento, vamos a conocer el elemento más característico de LOGO: la tortuga.

En los programas a realizar, tendremos una tortuga que moviéndonos a nuestro antojo, irá describiendo distintos gráficos.

Veamos, a continuación, cuáles son estas órdenes y cómo utilizarlas para que la tortuga nos obedezca.

3.1. INSTRUCCION "ADELANTE".

Esta instrucción provoca que la tortuga avance el número de pasos que especifiquemos.

EJEMPLO:

adelante 40

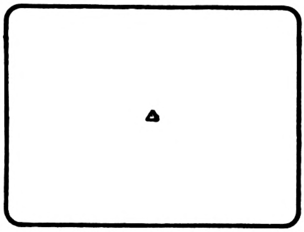
La tortuga avanzará 40 pasos en la dirección en que se encontraba:

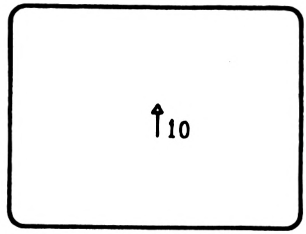


Con la instrucción adelante 40, la tortuga avanza 40 pasos.

EJEMPLO:

adelante 10





Con la instrucción adelante 10, la tortuga ha avanzado 10 pasos.

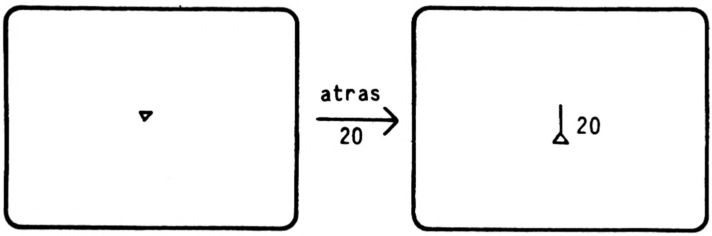
3.2. INSTRUCCION "ATRAS".

La instrucción atras permite a la tortuga retroceder un número de pasos que nosotros especificamos.

EJEMPLO:

atras 20

La tortuga retrocederá 20 pasos.



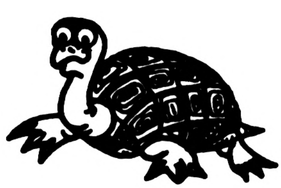
Con la instrucción atras 20, la tortuga ha retrocedido 20 pasos.

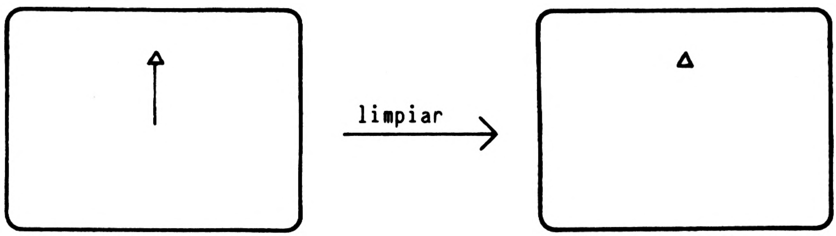
EJERCICIO:

Escribe una instrucción para que la tortuga retroceda 10 pasos.

3.3. INSTRUCCION "LIMPIAR".

Esta instrucción borra todos los dibujos realizados por la tortuga.





Con la instrucción limpiar se borran los dibujos de la tortuga.

EJEMPLO:

adelante 20

adelante 10

limpiar

Con las instrucciones anteriores la tortuga avanza 30 pasos y a continuación borra el sendero que ha ido marcando al avanzar.

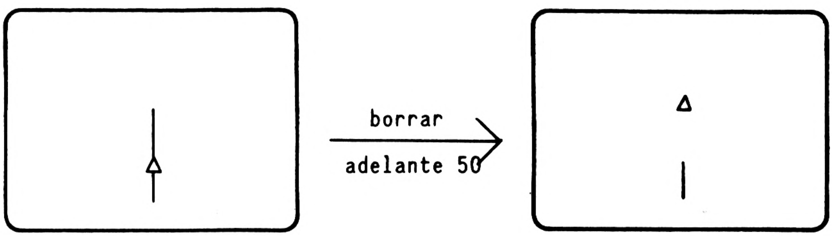
EJERCICIO:

Escribe una instrucción para que la tortuga avance 40 pasos y a continuación los borre mediante el uso de la instrucción borrar.

3.4. INSTRUCCION "BORRAR".

La instrucción borrar permite a la tortuga ir borrando las líneas por donde pasa.

EJEMPLO:



La tortuga ha borrado la linea por donde ha pasado.

Para que la tortuga deje de borrar, se escribe pluma visible y la tortuga volverá a pintar la trayectoria por donde pasa.

EJERCICIO:

Escribe una serie de instrucciones para que:

1. Se borre la pantalla.

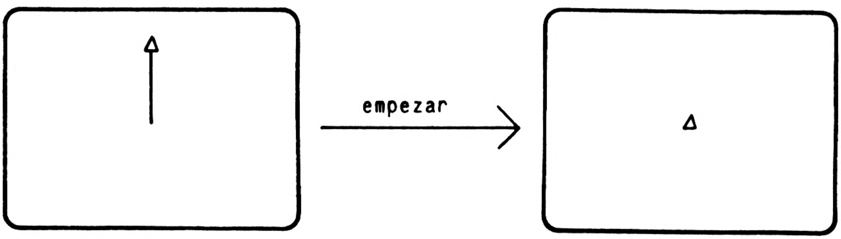
2. La tortuga avance 10 pasos.

3. La tortuga avance 20 pasos sin marcarlos.

4. La tortuga avance 5 pasos marcándolos.

3.5. INSTRUCCION "EMPEZAR".

Con el uso de esta instrucción, la tortuga volverá a su posición inicial (centro de la pantalla) borrando lo que había dibujado anteriormente.



Con la instrucción empezar, la tortuga vuelve a su casa (centro de la pantalla).

EJERCICIOS:

1. Escribe las instrucciones adecuadas para que la tortuga avance 20 pasos desde el centro de la pantalla.

   SOLUCION:

   empezar

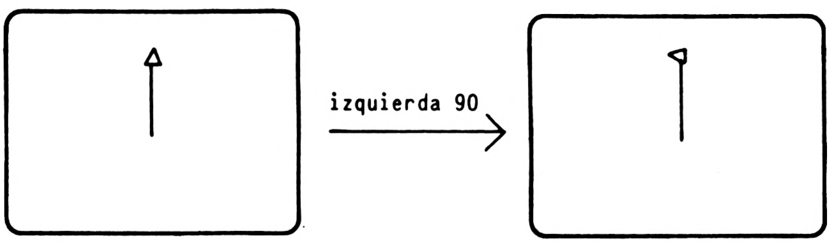
   adelante 20

2. Escribe las instrucciones necesarias para que la tortuga avance 40 pasos desde el centro de la pantalla.

3.6. INSTRUCCION "IZQUIERDA".

Con la instrucción izquierda seguida de un número, la tortuga gira hacia la izquierda el número de grados indicado.

EJEMPLO:



Con la instrucción izquierda 90 la tortuga ha girado 90 grados hacia la izquierda desde la posición en que se encontraba.

EJERCICIOS:

1. Escribe las instrucciones necesarias para que la tortuga:

   - Comience a caminar desde su posición original (centro de la pantalla).

   - Avance 40 pasos.

   - Gire 45 grados a la izquierda.

   - Avance 20 pasos.

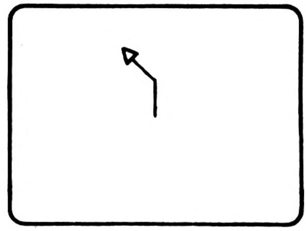
SOLUCION:

empezar

adelante 40

izquierda 45

adelante 20



Resultado final

2. Escribe las debidas instrucciones para que la tortuga:

- Comience a caminar desde su posición original.

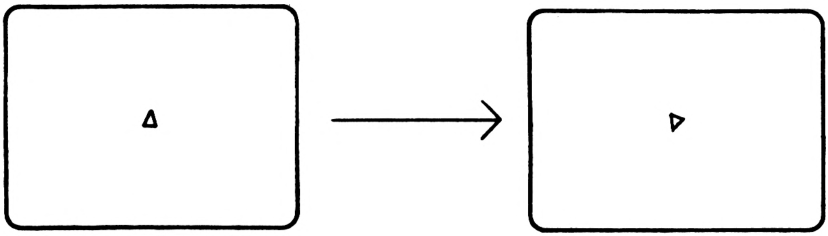
- Gire 90 grados a la izquierda.

- Avance 20 pasos.

—

3.7. INSTRUCCION "DERECHA".

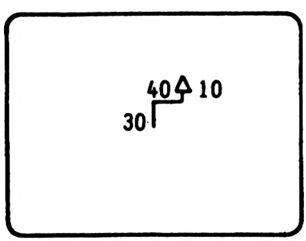
Con la instrucción "derecha" seguida de un número, la tortuga gira hacia la derecha el número de grados indicado.



Con la instrucción derecha 45, la tortuga gira 45 grados hacia la derecha.

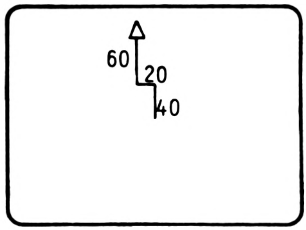
EJERCICIOS:

1. Escribe un conjunto de instrucciones que permitan a la tortuga realizar el dibujo de la figura siguiente:



SOLUCION: empezar
adelante 30
derecha 90
adelante 40
izquierda 90
adelante 10

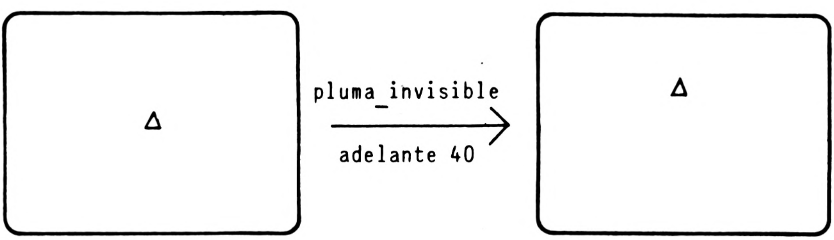
2. Escribe una serie de instrucciones para que la tortuga dibuje el gráfico de la figura.



3.8. INSTRUCCION "PLUMA_INVISIBLE".

Después de ejecutar esta instrucción, la tortuga, cuando se desplaza, no deja rastro.

EJEMPLO:



La tortuga avanza 40 pasos sin pintar.

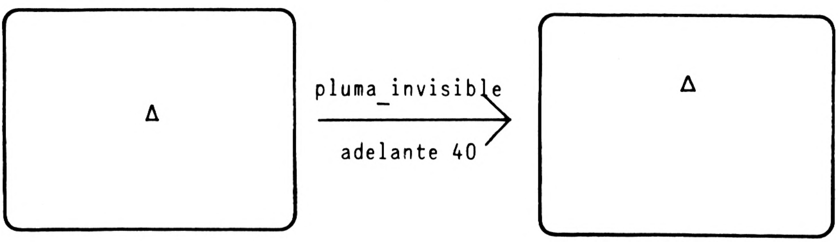
EJERCICIO:

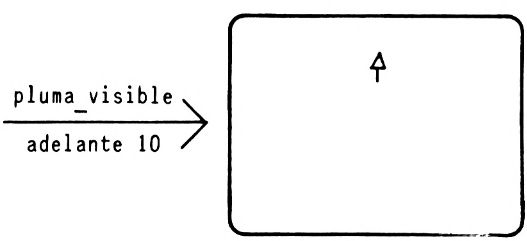
Escribe las instrucciones necesarias para que la tortuga avance 20 pasos, desde su posición inicial, sin marcarlos.

3.9. INSTRUCCION "PLUMA_VISIBLE".

Una vez ejecutada esta instrucción, la tortuga, cuando avance, irá describiendo el sendero que recorre.

EJEMPLO:





EJERCICIOS:

1. Escribe las instrucciones necesarias para que la tortuga dibuje una línea horizontal en la parte superior de la pantalla.

SOLUCION: empezar

pluma_visible

adelante 100

derecha 90

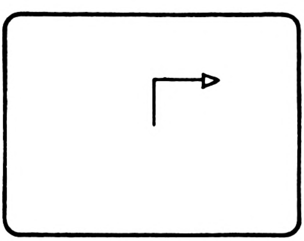
pluma_visible

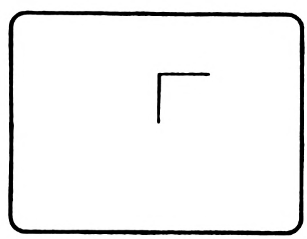
adelante 50

2. Escribe las instrucciones necesarias para que la tortuga dibuje una línea horizontal en la parte inferior de la pantalla.

3.10. INSTRUCCION ESCONDER_TORTUGA.

Con esta instrucción, la tortuga desaparece.





La tortuga desaparece después de escribir esconder_ tortuga.

EJERCICIOS:

1. Dibuja una línea de 20 pasos de tortuga, 45 grados a la izquierda a partir de su posición inicial, ocultando la tortuga una vez dibujada la línea.

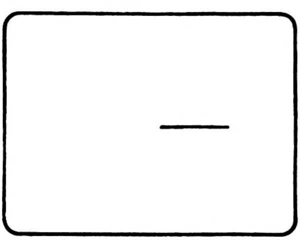
SOLUCION:

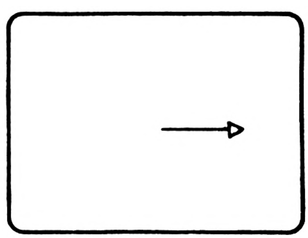
```
empezar
izquierda 45
adelante 20
esconder_tortuga
```

2. Dibuja una línea horizontal de 30 pasos de tortuga, ocultando la tortuga una vez dibujada la línea.

3.11. INSTRUCCION "VER_TORTUGA".

La tortuga reaparece, si estaba oculta, después de ejecutar esta instrucción.

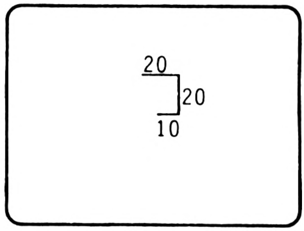




Con la instrucción "ver_tortuga", la tortuga vuelve a aparecer.

EJERCICIOS:

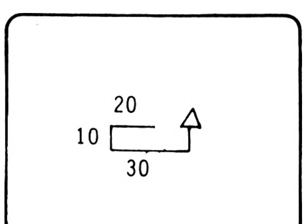
1. Realiza el dibujo de la figura haciendo que, una vez terminado, aparezca la tortuga.



SOLUCION:

```
empezar
esconder_tortuga
derecha 90
adelante 10
izquierda 90
adelante 20
izquierda 90
adelante 20
ver_tortuga
```

2. Realiza el dibujo de la figura haciendo, que una vez terminado, aparezca la tortuga.



3.12. INSTRUCCION "VUELVE_TORTUGA".

Con la instrucción "vuelve_tortuga", si la tortuga sale por algún lado de la pantalla, reaparece por el lado opuesto.



Mediante la instrucción "vuelve_tortuga", la tortuga, si desaparece por algún lado de la pantalla, reaparece por el lado opuesto.



En el ejemplo se observa que al salirse la tortuga por el lado superior de la pantalla reaparece por el lado inferior.

EJERCICIOS:

1. Escribe las instrucciones necesarias para que la tortuga dé 300 pasos hacia la derecha reapareciendo por el lado izquierdo de la pantalla.

   SOLUCION:
   ```
   empezar
   derecha 90
   vuelve_tortuga
   adelante 300
   ```

2. Escribe las instrucciones necesarias para que la tortuga avance 400 pasos hacia la izquierda reapareciendo por el lado derecho de la pantalla.

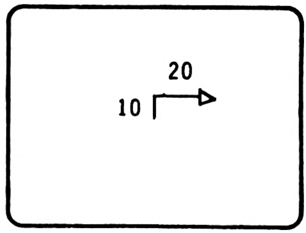
3.13. INSTRUCCION "COLOR_TORTUGA".

Cambia el color de la tortuga según la siguiente codificación:

Color_tortuga 0:	Tortuga de color del fondo y no se ve.
Color_tortuga 1:	Tortuga de color amarillo.
Color_tortuga 2:	Tortuga de color azul celeste.
Color_tortuga 3:	Tortuga de color rojo.

EJERCICIOS:

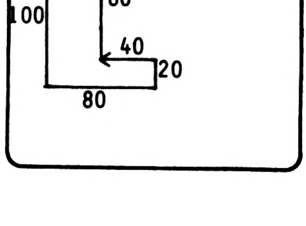
1. Realiza el gráfico de la figura en color amarillo.



SOLUCION:

```
empezar
color_tortuga 1
adelante 10
derecha 90
adelante 20
```

2. Realiza el gráfico de la figura en color azul celeste.

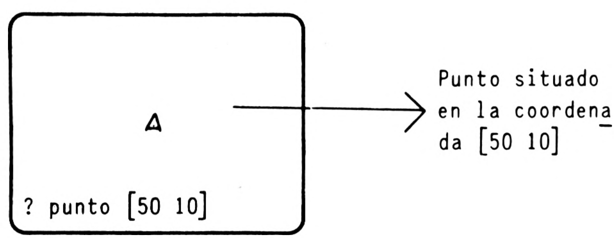


3.14. INSTRUCCION "PUNTO".

La instrucción "punto" dibuja un punto en la coordenada especificada. El primer parámetro es el valor del eje de abscisas y el segundo es el valor en el eje de coordenadas.

El punto central es el punto [0 0].

EJEMPLO:



Con la instrucción punto [50 10] , el ordenador pinta un punto en la coordenada [50 10]

EJERCICIO:

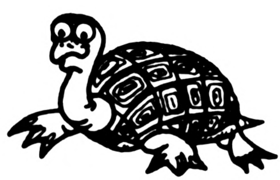
Dibuja un punto en el centro de la pantalla.

3.15. INSTRUCCION "PANTALLA_DE_GRAFICOS".

La instrucción "pantalla_de_gráficos" reserva la pantalla completa para el desplazamiento de la tortuga.

3.16. INSTRUCCION "PANTALLA_DE_TEXTO".

La instrucción "pantalla_de_texto" reserva la pantalla completa para introducir instrucciones.



RESUMEN DEL CAPITULO 3

* Instrucciones relacionadas con los gráficos de la tortuga:

- adelante:	La tortuga avanza en la dirección que se encuentre el número de pasos que se indiquen.
- atras:	La tortuga retrocede en la dirección en que se encuentra el número de pasos que se indican.
- limpiar:	Se borran todos los dibujos realizados por la tortuga.
- borrar:	La tortuga borra todas las líneas que se encuentran a su paso.
- empezar:	Se borran todos los dibujos realizados por la tortuga y ésta vuelve a su posición inicial (centro de la pantalla).
- izquierda:	La tortuga gira hacia la izquierda el número de grados que se indiquen.
- derecha:	La tortuga gira hacia la derecha el número de grados que se indiquen.
- pluma_invisible:	La tortuga deja de marcar el sendero por donde pasa.

- pluma_visible: La tortuga vuelve a marcar el sendero por donde pasa.

- esconder_tortuga: La tortuga desaparece.

- ver_tortuga: La tortuga reaparece.

- vuelve_tortuga: La tortuga reaparece por el lado opuesto al que se marchó.

- color_tortuga: La tortuga cambia de color.

- punto: La tortuga dibuja un punto en la pantalla.

- pantalla_de_gráficos: Se reserva la pantalla completa para gráficos.

- pantalla_de_texto: Se reserva la pantalla completa para texto.

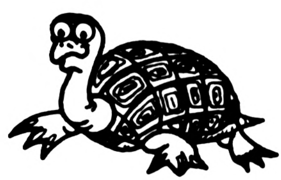
ACTIVIDADES

1. Escribe las instrucciones necesarias para dibujar un cuadrado.

2. Escribe las instrucciones necesarias para dibujar un rectángulo.

3. Escribe las instrucciones necesarias para dibujar un rombo.

4. Haz que la tortuga dibuje una cruz.



CAPITULO 4

Operaciones Aritméticas

Con LOGO podemos realizar las operaciones de:

* sumar
* restar
* multiplicar
* dividir

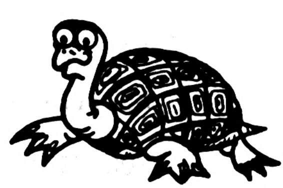
4.1. ADICION O SUMA.

Para realizar sumas se utiliza el signo "+". Podemos ver el resultado de una suma mediante la instrucción "escribir".

EJEMPLO:

Escribir "3 + 5

El ordenador escribirá después de teclear la instrucción anterior un ocho.



EJERCICIO:

Realiza la suma siguiente:

125 + 70 + 43

4.2. DIFERENCIA O RESTA.

Para realizar una resta se utiliza el signo "-"

EJEMPLO:

escribir "8 - 2

El ordenador escribirá, después de teclear la instrucción anterior, 6.

EJERCICIO:

Realiza la resta siguiente:

137 - 51

4.3. MULTIPLICACION.

El signo de la multiplicación es "*".

EJEMPLO:

Realiza el producto 5 * 4

SOLUCION: escribir "5 * 4

En el ejemplo anterior, el ordenador escribirá en la pantalla,después de ejecutar la instrucción, el número 20.

EJERCICIO:

Realizar el producto 3 * 20

4.4. DIVISION.

El signo de la división es "/".

EJEMPLO:

Realiza la división 8 / 4

SOLUCION: escribir "8 / 4

El ordenador escribirá después de teclear la instrucción anterior el número 2.

EJERCICIO:

Realiza la división 25 / 5

RESUMEN DEL CAPITULO 4

* Con LOGO podemos utilizar el ordenador como una calculadora.

* Se pueden realizar las operaciones de adición, sustracción, multiplicación y división.

CAPITULO 5
Concepto de procedimiento

5.1. CONCEPTO.

Hasta ahora hemos visto instrucciones con las que podemos dar órdenes sencillas. A estas instrucciones las podemos llamar primitivas, porque son las que componen el lenguaje LOGO.

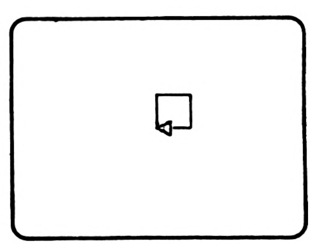
Uniendo varias primitivas, podemos crear instrucciones más complejas, pero que operan igual que las primitivas. A estas instrucciones las llamaremos procedimientos.

5.2. COMO CREAR UN PROCEDIMIENTO.

Para crear un procedimiento, escribiremos la palabra to, y a continuación, el nombre del procedimiento. Después se escriben las instrucciones del procedimiento, y por último, la palabra end.

Supongamos que queremos realizar un procedimiento que dibuje un cuadrado:

```
to cuadrado
adelante 30
derecha 90
adelante 30
derecha 90
adelante 30
derecha 90
adelante 30
end
```

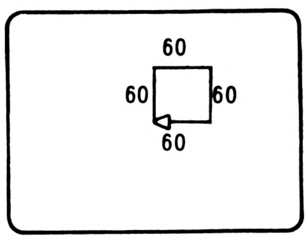


Cuadrado de 30 pasos de tortuga

El procedimiento del ejemplo anterior dibuja un cuadrado de 30 pasos de tortuga escribiendo la palabra cuadrado.

EJERCICIO:

Escribe un procedimiento que dibuje un cuadrado de 60 pasos de tortuga de lado.



5.3. COMO LISTAR LOS PROCEDIMIENTOS CREADOS

Tecleando la instrucción listar nos aparece en pantalla el conjunto de nombres de procedimientos que tenemos creados (junto con las instrucciones en castellano que son procedimientos que traducen las instrucciones del LOGO en inglés a castellano).

EJEMPLO:

```
listar
```

Después de escribir esta instrucción nos aparecerán los procedimientos que hemos realizado.

5.4. COMO LISTAR UN PROCEDIMIENTO.

Si queremos ver las instrucciones que componen un procedimiento, escribiremos lista_procedimiento seguido del nombre que tenga el procedimiento.

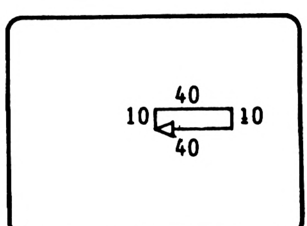
EJERCICIOS:

1. Dibuja un rectángulo de 40 pasos de largo y de 10 pasos de ancho listando a continuación el procedimiento.

SOLUCION:

```
to rectangulo
empezar
adelante 10
derecha 90
adelante 40
derecha 90
adelante 10
derecha 90
adelante 40
end
```

```
rectangulo
lista_procedimiento "rectangulo
```



El procedimiento rectángulo dibuja un rectángulo de 40 pasos de base por 10 pasos de altura.

2. Dibuja un rectángulo de 30 pasos de largo y 15 pasos de ancho, y a continuación, lista el procedimiento.

5.5. COMO BORRAR UN PROCEDIMIENTO.

Para borrar un procedimiento de memoria se escribe la instrucción er seguida del nombre del procedimiento.

También se pueden borrar del mismo modo las instrucciones en castellano que no se vayan a utilizar.

EJEMPLO:

```
er "pluma_visible
```

Con la instrucción del ejemplo, se borra la instrucción pluma_visible.

5.6. COMO MODIFICAR UN PROCEDIMIENTO.

Un procedimiento se modifica escribiendo la instrucción editar seguida del nombre del procedimiento. A continuación, nos aparecerán en pantalla las instrucciones que componen el procedimiento y podremos modificar las que deseemos utilizando los cursores (↑ → ↓ ←).

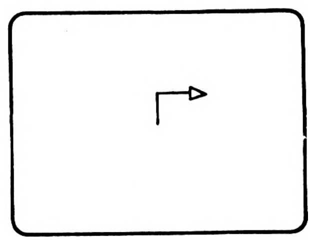
Una vez modificado el procedimiento, se pulsa la tecla <COPY> y el procedimiento queda grabado en memoria.

EJERCICIO:

Crea un procedimiento para dibujar un ángulo de 90 grados y después modifícalo para que dibuje un ángulo de 45 grados.

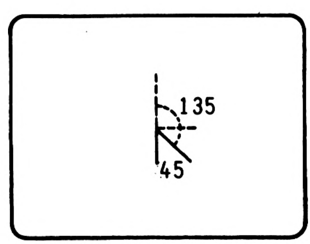
SOLUCION:

```
to angulo
empezar
adelante 30
derecha 90
adelante 30
end
```



El procedimiento ángulo dibuja un ángulo de 90 grados.

```
editar angulo
angulo
empezar
adelante 30
derecha 135
adelante 30
end
```



Modificando la instrucción derecha 90 por derecha 135 dibujamos un ángulo de 45 grados.

RESUMEN DEL CAPITULO 5

* Con los procedimientos podemos encadenar instrucciones y utilizarlas como si fueran instrucciones simples.

* Composición de un procedimiento:

```
to "nombre del procedimiento

____ }
     }          cuerpo del procedimiento
____ }

end
```

* Los procedimientos con variables permiten que puedan variar las cantidades que intervienen en el cuerpo del procedimiento.

* listar:	sale por pantalla el nombre de los procedimientos que están en memoria.
* lista_procedimiento:	salen por pantalla las instrucciones del procedimiento especificado.
* editar:	permite modificar un procedimiento grabado previamente.

ACTIVIDADES

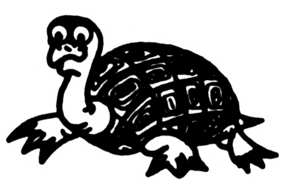
1. Escribe un procedimiento que dibuje un triángulo.

2. Escribe un procedimiento que dibuje un cuadrado de lado variable.

3. Programa un procedimiento que realice un rectángulo de 50 pasos de tortuga de largo por 30 pasos de ancho.

4. Modifica el procedimiento del ejercicio anterior para dibujar un rectángulo de 60 pasos de largo por 40 pasos de ancho.

5. Saca por la pantalla los procedimientos que tienes en memoria.

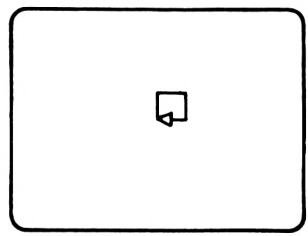


CAPITULO 6

Concepto de bucle

Supongamos que queremos construir un cuadrado de 20 pasos de tortuga:

```
to cuadrado
adelante 20
derecha 90
adelante 20
derecha 90
adelante 20
derecha 90
adelante 20
derecha 90
end
```



Como ves, para dibujar un cuadrado es necesario repetir 4 veces las órdenes siguientes:

Que la tortuga avance 20 pasos Que gire 90 grados	}	A estos dos pasos que se repiten, se les llama bucle.

Sería más cómodo escribir una sola vez estas dos instrucciones y que fuesen repetidas 4 veces.

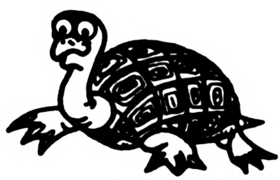
Un bucle es la repetición de un conjunto de instrucciones, mientras se cumple alguna condición. (En el ejemplo anterior, la condición sería que hasta que no se repitan por cuarta vez las instrucciones indicadas no se salga del bucle).

6.1. INSTRUCCION "REPETIR".

Con la instrucción repetir, vamos a poder programar bucles. Tomemos de nuevo el ejemplo del cuadrado. Si escribimos:

```
repetir 4 [adelante 20 derecha 90]
```

La tortuga repetirá 4 veces las instrucciones que están entre corchetes dibujando un cuadrado.



EJERCICIO:

Escribe un procedimiento para dibujar una circunferencia.

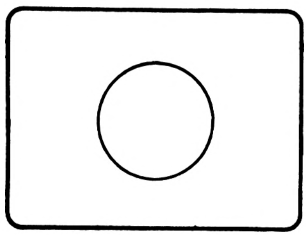
SOLUCION:

Podemos hacer que la tortuga dibuje una circunferencia haciéndola avanzar cortos espacios y obligándola a girar repetidamente.

Cuando haya girado 360 grados, habrá descrito una circunferencia.

El procedimiento es el siguiente:

```
to circunferencia
empezar
repetir 360 [adelante 3 derecha 1]
end
```



La tortuga, con este procedimiento, dibuja un polígono de 360 lados de 3 pasos de tortuga cada uno. (Figura parecida a una circunferencia que se asemeja a un polígono de infinito número de lados).

RESUMEN DEL CAPITULO 6

* Los bucles permiten que se repitan un conjunto de instrucciones un determinado número de veces, mientras se cumple una condición.

* La instrucción repetir hace que se repitan una o varias instrucciones el número de veces que se especifica.

ACTIVIDADES

1. Escribe un procedimiento para dibujar un pentágono.

2. Dibuja un mosaico mediante la repetición de un procedimiento que dibuje un cuadrado.

CAPITULO 7
Tratamiento de listas

Las listas son, al igual que la tortuga, elementos característicos del lenguaje LOGO. Existen instrucciones especiales de tratamiento que ofrecen una potencia superior en LOGO a la de otros lenguajes de programación.

7.1. CONCEPTO DE LISTA.

Una lista es una cadena de caracteres sobre la que podemos realizar una serie de operaciones que vamos a ver a continuación.

7.2. INSTRUCCION "UNE".

La instrucción une hace que se junten dos listas.

EJEMPLO:

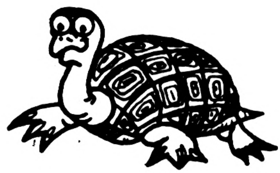
```
une "ven "tana
```

LOGO nos dará como resultado de la instrucción anterior la unión de la lista "ven" con la lista "tana" escribiendo:

```
ventana
```

EJERCICIO:

Escribe la palabra "campeon" a partir de la lista "cam y de la lista "peon .



7.3. INSTRUCCION "SIN_PRIMERA".

La instrucción sin_primera seguida de una variable y de una lista quita la primera letra de la lista e introduce el resto de caracteres en la variable.

EJEMPLO:

```
sin_primera  "v  "peseta
```

Si ahora tecleamos escribir :v nos aparecerá en pantalla:

```
eseta
```

EJERCICIOS:

1. Escribe un procedimiento para que introduciéndole al ordenador un nombre, nos dé como resultado todas las letras que componen el nombre menos la inicial.

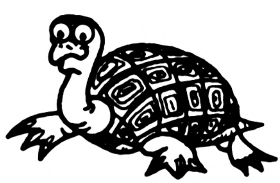
SOLUCION:

```
to nombre
escribir "dame_tu_nombre
entrada "nombre
sin_primera "solucion :nombre
escribir :solucion
end
```

2. Escribir un programa para que al introducir en el ordenador el nombre de un país, nos dé como resultado todas las letras que componen el nombre menos la inicial.

7.4. INSTRUCCION "SIN_ULTIMA".

La instrucción sin_última seguida de una variable y de una palabra (o lista) quita la última letra de la palabra e introduce el resto de caracteres en la variable.



EJEMPLO:

```
sin_ultima "w "telefono
```

Para ver el contenido de la variable w después de ejecutar la anterior instrucción, hay que introducir la orden:

```
escribir :w
```

A continuación, podemos observar que en pantalla aparece:

```
telefon
```

<u>EJERCICIOS:</u>

1. Escribe un programa que al introducir el nombre de un río, nos de como resultado todas las letras que componen el nombre del río menos la última.



SOLUCION:

```
to rio
escribir "dame_rio
entrada "nombre_rio
sin_ultima "solucion :nombre
                              -rio
escribir :solucion
end
```

2. Escribe un programa que introduciendo en el ordenador el nombre de una ciudad nos dé como resultado todas las letras que componen el nombre menos la última.

RESUMEN DEL CAPITULO 7

* Una lista es una cadena de caracteres (se exceptúa el carácter blanco).

* une: junta dos líneas en una sola.

* sin_primera: elimina la primera letra de la lista.

* sin_última: elimina la última letra de la lista.

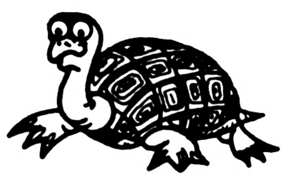
ACTIVIDADES

1. Escribe un procedimiento para suprimir la primera letra de la palabra libro.

2. Escribe un procedimiento para eliminar la última letra de la palabra asignatura.

3. Une en una palabra las listas "inter" y "nacional" para que resulte la palabra "internacional".

4. Escribe un procedimiento que escriba todas las letras de la palabra piso menos las dos últimas.



CAPITULO 8

Operaciones con disco

Las operaciones básicas para trabajar con disco son:

- Cargar
- Salvar
- Sacar el directorio de un disco
- Concluir una sesión

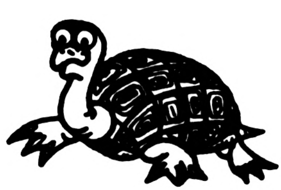
8.1. CARGAR UN DISCO.

La instrucción cargar seguida del nombre de un fichero, hace que se cargue el fichero en memoria y podamos trabajar con los procedimientos que contenga.

EJEMPLO:

```
cargar "fich
```

Al cabo de unos segundos se visualizarán en pantalla los nombres de los procedimientos que pertenecen al fichero "fich". (Para ello debe haber sido grabado previamente el fichero "fich").



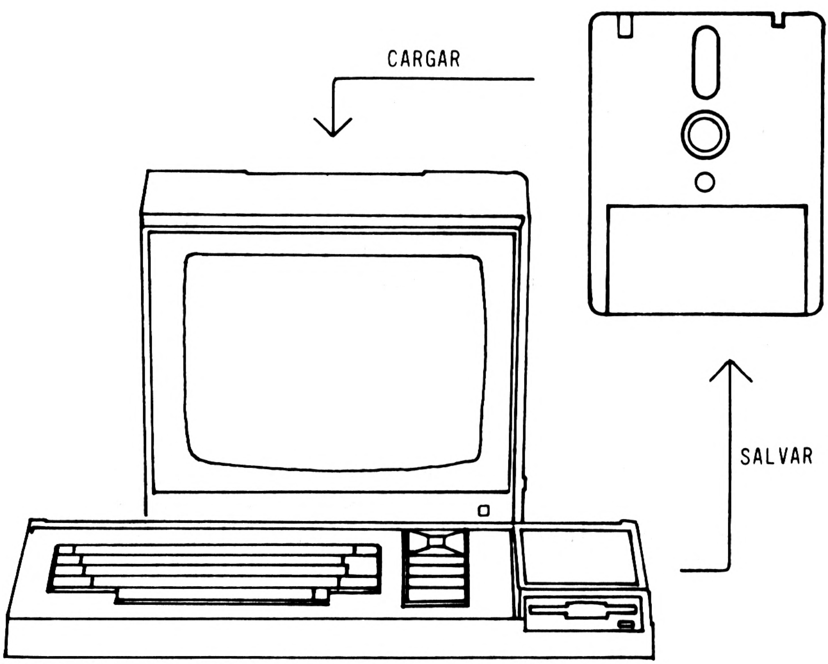
8.2. SALVAR EN DISCO.

La instrucción salvar es lo opuesto de la instrucción de cargar. Provoca que los procedimientos que hayamos realizado pasen al disco con el nombre que deseemos poner.

EJEMPLO:

salvar "nombre

Creará un fichero en disco llamado "nombre" con los procedimientos que hayamos construido.



EJERCICIO:

Dibujar un rectángulo de 30 pasos de tortuga de largo y de 15 pasos de ancho y grabarlo con el nombre rect.

8.3. DIRECTORIO DE UN DISCO.

El directorio de un disco nos dá el nombre de los ficheros que tengamos en ese disco.

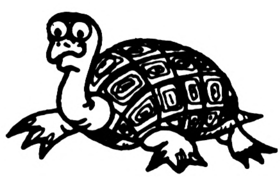
Para sacar el directorio de un disco introduciremos el disco en el drive, y escribiremos:

```
directorio
```

A continuación, veremos los ficheros en LOGO del disco.

EJERCICIO:

Haz que el ordenador liste por la pantalla los ficheros en LOGO de tu disco.



8.4. FINALIZACION DE UNA SESION.

Cuando queramos acabar una sesión de LOGO, grabaremos en disco los procedimientos que nos interesen (según lo explicado en el apartado 8.2) y escribiremos:

```
adios "ordenador
```

saliéndonos del lenguaje LOGO.

RESUMEN DEL CAPITULO 8

Las instrucciones que permiten trabajar con disco son:

* cargar: carga un fichero de disco y lo introduce en memoria.

* salvar: graba los procedimientos de memoria en disco.

* adios "ordenador: para abandonar una sesión de LOGO.

CAPITULO 9

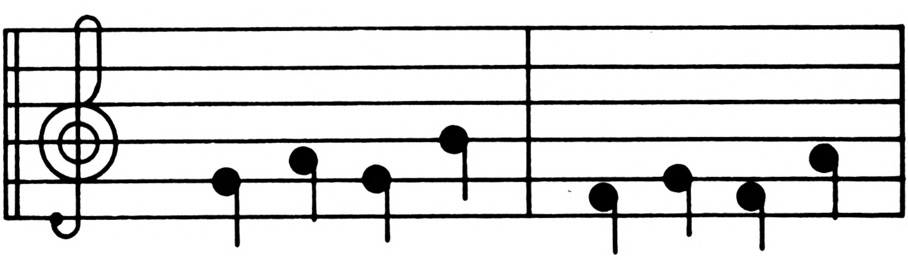
Sonidos

Para realizar sonidos con el ordenador utilizaremos la instrucción "sonido" seguida de cuatro números entre corchetes con el siguiente significado:

- El primero: Indica el número de canal (1-3).
- El segundo: Tono de la nota (cuanto más alto sea, mayor agudeza de la nota).
- El tercero: Duración de la nota.
- El cuarto: Volumen (valores 1-15).

EJEMPLO:

sonido [1 100 20 15]



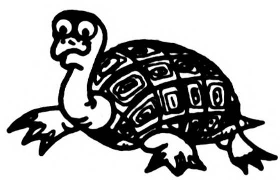
EJERCICIOS:

1. Escribe una instrucción para que el ordenador produzca un sonido grave.

   SOLUCION:

   sonido [1 800 40 15]

2. Haz que el ordenador emita un sonido agudo.



CAPITULO 10

Comandos de gestión de memoria

Las instrucciones de LOGO para trabajar con la memoria son:

- espacio
- deja_espacio

10.1. INSTRUCCION "ESPACIO".

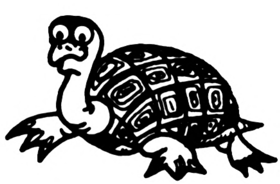
Si escribimos espacio nos aparece el número de nodos que nos quedan libres en memoria. (Un nodo es una pequeña división de la memoria).

EJERCICIO:

Averigua el número de nodos libres.

SOLUCION:

```
espacio
```



10.2. INSTRUCCION "DEJA_ESPACIO".

Después de llevar un determinado período de tiempo trabajando, podemos escribir el comando deja_espacio, que liberará espacio en memoria.

EJEMPLO:

```
espacio
```

(El ordenador nos dirá el espacio de memoria que le queda libre).

```
libera_espacio
```

(Después de unos segundos, el ordenador habrá aprovechado mejor el espacio de memoria dejando más nodos libres).

EJERCICIO:

Averigua el espacio libre de memoria, ampliando a continuación dicho espacio.

SOLUCION:

```
espacio
libera_espacio
espacio
```

RESUMEN DE LOS CAPITULOS 9 Y 10

* Se pueden emitir sonidos mediante la instrucción sonido [a b c d], donde:

  a: número de canal

  b: tono

  c: duración

  d: volumen

* En LOGO tenemos instrucciones para operar con la memoria disponible:

  - espacio: Escribe el número de nodos que tenemos libres.

  - libera_espacio: Proporciona un mayor número de nodos de memoria disponibles.

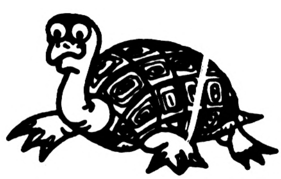
CAPITULO 11

Generación de números aleatorios

Una sucesión de números aleatorios es una serie de números que no siguen una secuencia predefinida y cada vez que los hallamos obtendremos una sucesión distinta.

11.1. INSTRUCCION "ALEATORIO".

La instrucción aleatorio va seguida de un número y de una variable. El número aleatorio generado será menor que el número que sigue a la instrucción y va a ser el contenido de la variable que va a continuación.



EJEMPLO:

aleatorio 400 "c

Esta instrucción genera un número aleatorio menor que 400 y lo introduce en la variable c.

Si queremos ver el número que ha generado, escribiremos:

```
escribir :c
```

Si ahora repetimos la instrucción:

```
aleatorio 400 "c
```

podemos observar que c contiene otro valor.

Cada vez que ejecutemos una instrucción aleatorio, el resultado será distinto.

<u>EJEMPLO:</u>

Genera un número aleatorio menor que 1000.

SOLUCION:

```
aleatorio 1000 "numero
escribir :numero
```

<u>EJERCICIO:</u>

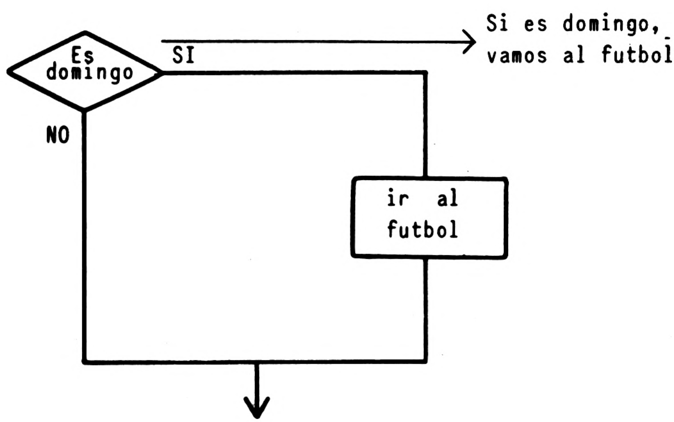
Genera un número aleatorio menor que 500.

CAPITULO 12

Control secuencia Instrucción «si»

Cuando queramos que una acción se efectúa o no dependiendo de una determinada condición, utilizaremos la instrucción si seguida de la condición.

Previamente habremos introducido en una variable llamada instrucción la acción que hay que realizar si se cumple la condición. De no cumplirse la condición, se continúa con la ejecución de la instrucción siguiente.



EJEMPLO:

Supongamos que si una determinada variable v tiene un valor menor que 5, la tortuga deberá avanzar 20 pasos.

Definimos, en el ejemplo, una condición y una acción:

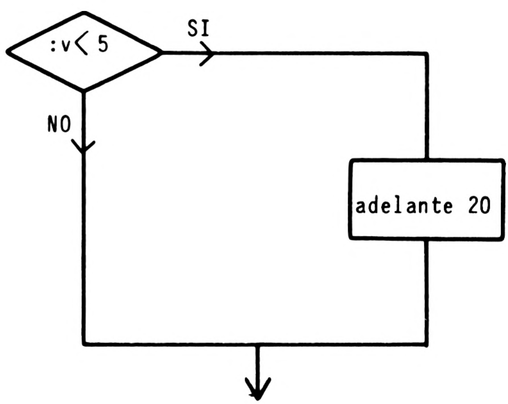
Condición ⟶ $:v < 5$

Acción ⟶ adelante 20

SOLUCION:

```
haz "instrucción [adelante 20]
entrada "v
si (:v< 5)
```

Si introducimos en la variable v, mediante la instrucción entrada, un número menor que 5, se cumple la condición y, por tanto, se ejecuta la instrucción avanzando la tortuga 20 pasos.



EJERCICIO:

Escribe un programa, que en función de que una variable sea o no igual a 6, haga o no retroceder 20 pasos a la tortuga.

CAPITULO 13

Operadores lógicos

Los operadores lógicos permiten relacionar condiciones. Estos operadores son:

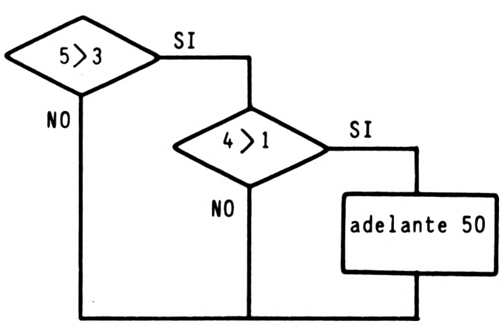
- Operador and
- Operador or
- Operador not

13.1. OPERADOR "AND".

El operador and hace que al evaluarse dos condiciones, se dé como resultado cierto si se cumplen las dos condiciones.

EJEMPLO:

```
haz "instruccion [adelante 50]
si (and (5 >3) (4 >1))
```



Al cumplirse las dos condiciones, 5 es mayor que 3 y 4 es mayor que 1, la tortuga avanza 50 pasos.

EJERCICIO:

Escribir un programa, en el cual, a partir de la introducción de un mes y de un día, nos diga si la fecha introducida es Navidad.

SOLUCION:

```
escribir "numero_de_mes
entrada "nm
escribir "dia_de_mes
entrada "dm
haz "instruccion [escribir "Es_Navidad]
si (and (:nm = 12) (:dm = 25))
```



13.2. OPERADOR OR.

El operador OR, al evaluarse dos condiciones, dá como resultado cierto si se cumple alguna de las dos condiciones o bien si se verifican las dos.

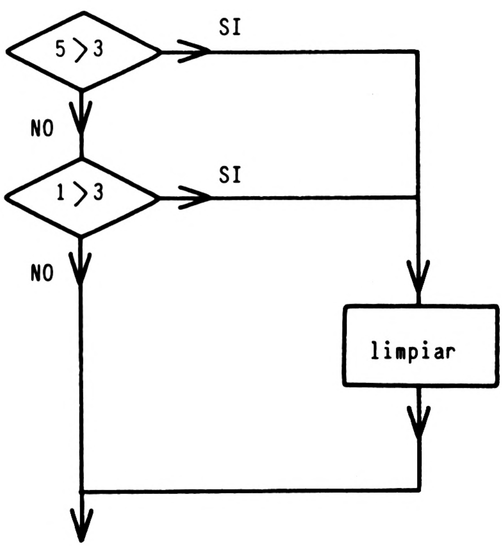
EJEMPLO:

```
haz "instruccion [limpiar]
si (or (5 >3) (1 >3))
```

En este ejemplo, se ejecuta la instrucción de limpiar si se cumple alguna de estas dos condiciones:

1. Que 5 sea mayor que 3.
2. Que 1 sea mayor que 3.

Al cumplirse, al menos una de las dos condiciones, -la primera se cumple- se ejecuta la instrucción de limpiar y el ordenador borrará el contenido de la pantalla.



Al cumplirse la condición (5 >3), el ordenador borra la pantalla.

EJERCICIO:

Para entrar a presenciar un espectáculo deportivo, los menores de 18 años no pagan entrada y los mayores deben pagar 1000 pesetas para poder entrar.

Escribe un programa en el que a partir de la edad y del dinero que posea una persona, el ordenador escriba si puede o no presenciar el espectáculo.

SOLUCION:

Para que se pueda ver el espectáculo, existen dos condiciones:

- Ser menor de 18 años.
- Tener al menos 1000 pesetas, es decir, más de 999 pesetas.

Cumpliéndose una de las dos condiciones, se ejecutará la acción:

El ordenador escribirá en pantalla que puede presenciar el espectáculo.



Para resolver el problema, escribiremos el siguiente procedimiento:

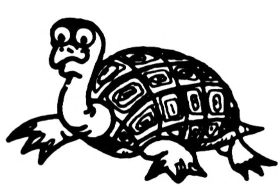
```
to taquilla
haz "instruccion [escribir "puede_
presenciar_el_espectaculo]
escribir "edad
entrada "ed
escribir "cuanto_dinero_tiene
entrada "di
si ((:ed< 18) or (:di >1000))
end
```

13.3. OPERADOR NOT.

El operador not seguido de una condición hace que al evaluarse ésta, se dé como resultado cierto si la condición era falsa o viceversa.

EJERCICIO:

Introduce un número en una variable llamada p de modo que si no es mayor que 10, el ordenador escriba que se ha introducido un número de una cifra.



RESUMEN DE LOS CAPITULOS 11, 12 Y 13

* aleatorio: Genera un número aleatorio.

* si: Produce una bifurcación condicional. (Si se cumple una condición, se ejecuta una instrucción).

* Los operadores lógicos relacionan condiciones dando como resultado cierto o falso.

  - Operador and: Dá como resultado cierto si se cumple alguna de las condiciones que intervienen.

  - Operador or: Dá como resultado cierto si se cumple alguna de las condiciones que intervienen.

  - Operador not: Dá como resultado cierto si no se cumple la condición que se evalúa.

ACTIVIDADES

1. Un alumno va todos los días de la semana al colegio, excepto los sábados y los domingos.

   Escribe un procedimiento que diga si esa persona debe asistir a clase a partir de la introducción en el ordenador de un determinado día.

2. A Luisito, sus padres, le han prometido un ordenador de regalo si aprueba las matemáticas o el lenguaje.

   Escribe un procedimiento que diga si Luisito tendrá o no ordenador a fin de curso en función de las notas que saque.

CAPITULO 14
Recursividad

La recursividad es una propiedad mediante la cual un procedimiento se llama a sí mismo.

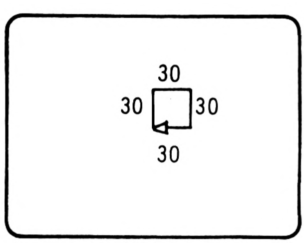
14.1. UTILIDAD.

La recursividad es muy útil puesto que con un solo procedimiento se pueden generar efectos sorprendentes.

Supongamos que queremos realizar un dibujo compuesto de cuadrados que van girando cada uno con relación al siguiente un ángulo de 30 grados.

En primer lugar, definiremos el procedimiento cuadrado:

```
to cuadrado
repetir 4 [adelante 30 derecha 90]
end
```

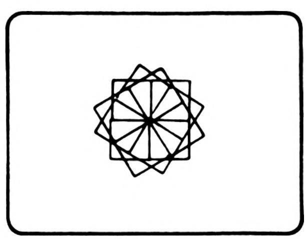


Este procedimiento, por sí solo, nos dibuja un cuadrado.

Si queremos dibujar infinitos cuadrados separados unos de otros 30 grados, dibujaremos un cuadrado, giraremos a la derecha 30 grados y volveremos otra vez a dibujar un cuadrado mediante la llamada cuadrado. Este proceso se repetirá indefinidamente.

```
to cuadrado
repetir 4 [adelante 30 derecha 90]
derecha 30
cuadrado
end
```

El procedimiento recursivo cuadrado, dibuja este cuadrado basado en cuadrados que van girando:



Como este programa no acaba nunca, si queremos cortar su ejecución, pulsaremos la tecla ESC.

Podíamos haber controlado su final desde el programa con una instrucción condicional de forma que si se cumpliese una condición se llamaría al procedimiento y de no ser así se acabaría el procedimiento.

EJEMPLO:

```
haz "instruccion [cuadrado]
haz  "veces 0
to cuadrado
repetir 4 [adelante 30 derecha 90]
derecha 30
haz "veces  :veces +1
si (:veces <12)
end
```

Con el nuevo procedimiento la variable veces va aumentano en una unidad cada vez que se realiza el cuadrado. Una vez que se han dibujado 12 cuadrados ya no se cumple la condición:

:veces < 12

Y a partir de este momento concluye la ejecución del procedimiento dejando de dibujarse cuadrados.

EJERCICIO:

Programa un procedimiento recursivo para calcular el factorial de un número n:

$$n! = n*(n-1)*(n-2)*\dots 1$$

RESUMEN DEL CAPITULO 14

* La recursividad es la facultad que tienen ciertos lenguajes para que un procedimiento se llame a sí mismo.

* Con procedimientos recursivos se pueden generar potentes cálculos con pocas líneas de instrucciones.

* Si no se pone una condición para terminar el procedimiento, éste se ejecutará indefinidamente hasta que pulsemos la tecla<ESC>.

ANEXO 1

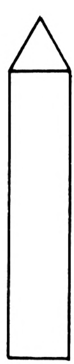
PROGRAMAS-EJEMPLO

1. Realiza un programa para calcular la media aritmética de un conjunto de cantidades.

```
to media
escribir "cuantas_cantidades
entrada "n
haz "suma 0
repetir :n [escribir "dime_un_numero
        entrada "p haz "suma :suma + :p]
haz "resultado :suma / :n
escribir "la_media_es
escribir :resultado
end

media
```

2. Realiza un programa que dibuje un lapicero:

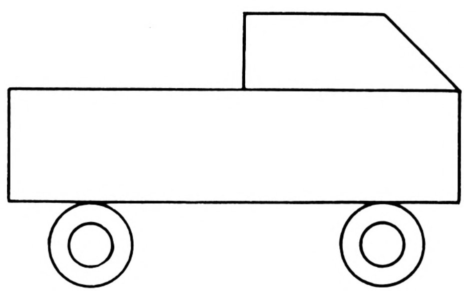


```
to triangulo :lado
adelante :lado
izquierda 120
adelante :lado
izquierda 120
adelante :lado
izquierda 120
end

to rectangulo :base :altura
repetir 2 [adelante :base izquierda 90
        adelante :altura izquierda 90]
end
```

```
to lapiz
pluma_invisible
empezar
adelante 100
pluma_visible
derecha 90
triangulo 50
derecha 90
rectangulo 200 50
end
lapiz
```

3. Realiza un programa para dibujar un camión.



```
to circulo :radio
repetir 24 [triangulo :radio derecha 15]
end

to ruedas
empezar
pantalla_de_graficos
pluma_invisible
derecha 90
adelante 150
derecha 90
adelante 100
pluma_visible
circulo 30
derecha 90
```

```
pluma_invisible
adelante 300
pluma_visible
circulo 30
derecha 90
end

to camion
ruedas
adelante 30
izquierda 90
adelante 50
derecha 180
rectangulo 400 80
adelante 400
izquierda 90
adelante 80
izquierda 45
adelante 70
izquierda 45
adelante 100
izquierda 90
adelante 50
end

camion
```

4. Escribir un programa que calcule la edad que tiene una persona a partir de su fecha de nacimiento y del año actual.

```
to edad
escribir "año_de_nacimiento
entrada "año
escribir "año_actual
entrada "aa
haz "ed :aa  -  :año
escribir "la_edad_es
escribir :ed
end
```

ANEXO 2

Instrucciones análogas del LOGO AMSTRAD a las del LOGO CAST:

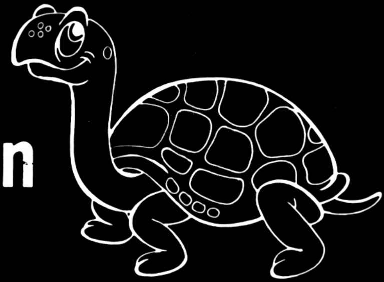
CAST	AMSTRAD
si	if
atras	bk
izquierda	lt
empezar	cs
borrar	pe
derecha	rt
ver	st
adelante	fd
listar	pots
cargar	load
salvar	save
haz	make
escribir	pr
espacio	nodes
une	word
adios	bye
punto	dot
sonido	sound
aleatorio	random
editar	ed
entrada	rq
directorio	dir
repetir	repeat
cuenta	count
pluma_visible	pd
pluma_invisible	pu
ver_tortuga	st
esconder_tortuga	ht
borrar_variable	ern
color_tortuga	setpc

pantalla_de_graficos	fs
vuelve_tortuga	wrap
sin_primera	bf
sin_ultima	bl
pantalla_de_texto	ts
listar_procedimiento	po
dejar_espacio	recycle

NOTA:

La utilización de una de las primitivas en LOGO AMSTRAD no coincide con las traducciones a LOGO CAST.

Aprende LOGO paso a paso con la Tortuga.



Entre los innumerables lenguajes de programación existentes, cada uno con una orientación específica, LOGO destaca por su caracter pedagógico.

Su creador Seymur Papert, desarrolló en el Instituto Tecnológico de Massachusets (M.I.T), al frente de un equipo de colaboradores expertos en investigación didáctica, un lenguaje que ha supuesto una auténtica revolución en las aplicaciones del ordenador a la enseñanza.

Aprende LOGO con AMSTRAD, utiliza la versión castellana de este lenguaje adaptada y traducida para los ordenadores CPC-6128, 664 y 464 por SPEN, S.A. Centro Educativo de Informática.

ISBN 84-86381-16-8

ra-ma editorial

CARRETERA DE CANILLAS, 144 - 28043 MADRID
TELEFS. (91) 200 97 46 - 47

Aprende LOGO con AMSTRAD

Ficheros en castellano